МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ «ВИТЕБСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ П. М. МАШЕРОВА»

Факультет математики и информационных технологий

Кафедра прикладного и системного программирования

КУРСОВАЯ РАБОТА

по дисциплине «Базы данных»

АНАЛИЗ ОСНОВНЫХ КАТЕГОРИЙ НЕРЕЛЯЦИОННЫХ БАЗ ДАННЫХ. РАЗАРАБОТКА СХЕМЫ ДОКУМЕНТА.

Царенко Антон Дмитриевич,

4 курс, 44 группа

Научный руководитель:

Новый Вадим Владимирович,

Старший преподаватель кафедры прикладного и системного программирования

Витебск, 2023

**Реферат**

Курсовая работа 17 стр., 8 рис., 5 источников.

НЕРЕЛЯЦИОННЫЕ БАЗЫ ДАННЫХ, NOSQL, MONGODB.

**Объект исследования** – технологии для работы с нереляционными базами данных.

**Предмет исследования** – алгоритмы проектирования схемы документа для нереляционных баз данных.

**Цель работы** – исследование основных категорий нереляционных баз данных, в частности, MongoDB, а также проектирование структуры документа для хранения информации о входе и выходе пользователей из учетных записей.

**Задачи**:

1. Выполнить анализ открытых источников информации, литературы в сфере нереляционных баз данных.
2. Ознакомиться с основными категориями нереляционных баз данных
3. Ознакомиться с особенностями MongoDB.
4. Спроектировать структуру документа.

**Методы исследования** – изучение документации, материалов научных и периодических изданий по теме исследования, существующих систем, общенаучные методы исследования (описание, анализ и т.д.).

**Теоретическая значимость и элементы новизны**: совершенствование знаний в области нереляционных баз данных.

**Практическая значимость**: спроектированная схема документа позволяет в дальнейшем продолжить разработку приложения для отсеживания информации о входе и выходе пользователей из учетных записей.

Содержание

[Введение 4](#_Toc153869856)

[1 Основные категории нереляционных баз данных 6](#_Toc153869857)

[2 MongoDB. Проектирование схемы документа 12](#_Toc153869858)

[2.1 Обоснование выбора и общая характеристика MongoDB 12](#_Toc153869859)

[2.2 Проектирование структуры документа 14](#_Toc153869860)

[Заключение 16](#_Toc153869861)

[Список использованных источников 17](#_Toc153869862)

Введение

Нереляционные базы данных, также известные как NoSQL (Not Only SQL), представляют собой альтернативу реляционным базам данных (SQL), которые основаны на традиционной модели таблиц и связей между ними. Нереляционные базы данных используют различные модели данных и подходы к хранению и организации информации, что позволяет эффективно работать с большими объемами данных и обеспечивать гибкость и масштабируемость.

К преимуществам нереляционных баз данных можно отнести:

1. Гибкость схемы данных: в отличие от реляционных баз данных, нереляционные базы данных не требуют строгой фиксированной схемы данных. Это означает, что можно добавлять, изменять или удалять поля в документах или объектах без необходимости обновления всей базы данных. Это особенно полезно в случаях, когда схема данных может меняться или когда данные имеют различные структуры;
2. Горизонтальное масштабирование: нереляционные базы данных обеспечивают лучшую горизонтальную масштабируемость, что означает, что их можно легко масштабировать на большое количество серверов или узлов. Это позволяет обрабатывать большие объемы данных и обеспечивать высокую производительность при распределенной нагрузке;
3. Гибкий формат данных: нереляционные базы данных поддерживают гибкие форматы данных, такие как JSON, XML или бинарные форматы, что делает их удобными для работы с полуструктурированными или неструктурированными данными. Это особенно полезно при работе с веб-данными, журналами, текстовыми документами и другими нестандартными форматами;

Важно отметить, что преимущества нереляционных баз данных зависят от конкретного сценария использования и требований проекта.

**Цель работы** – анализ основных категорий нереляционных баз данных, а также разработка схемы документа для хранения информации о входе и выходе пользователей из учетных записей.

Для достижения поставленной цели необходимо решить следующие **задачи**:

1. Выполнить анализ открытых источников информации, литературы в сфере нереляционных баз данных.
2. Ознакомиться с особенностями MongoDB
3. Составить схему документа.

**Методы исследования** – изучение документации, материалов научных и периодических изданий по теме исследования, существующих систем разработки редакторов, общенаучные методы исследования (описание, анализ, классификация, сравнение, аналогия).

Работа включает введение, 2 главы, заключение, список использованных источников. В первой главе предоставлено описание основных категорий нереляционных баз данных. Во второй главе приводится общая характеристика MongoDB, обоснованность ее выбора, а также проектирование схемы документа.

1 Основные категории нереляционных баз данных

К основным категориям нереляционных баз данных можно отнести следующие:

* Хранилища данных документов

Нереляционные базы данных, использующие хранилища данных документов, предоставляют гибкую модель хранения информации в виде документов. Документы обычно представлены в форматах, таких как JSON (JavaScript Object Notation) или BSON (Binary JSON), и могут содержать различные поля и структуры данных.

Как правило, документ содержит все данные для сущности. Элементы, составляющие сущность, зависят от конкретного приложения. Один документ может содержать сведения, которые в реляционной СУБД обычно распределяются по нескольким реляционным таблицам. Хранилище документов не требует, чтобы все документы имели одинаковую структуру. Такой свободный подход к форме обеспечивает большую гибкость.

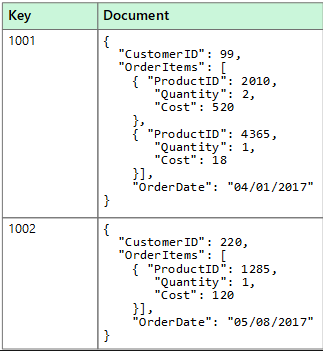


Рисунок 1 – хранилище данных документов.

Примеры нереляционных баз данных, основанных на хранилищах данных документов: MongoDB, Couchbase, RavebDB.

* Хранилище пар «ключ – значение»

Хранилище пар «ключ – значение» по сути представляет собой большую хэш-таблицу. Каждое значение сопоставляется с уникальным ключом, и хранилище ключей использует этот ключ для хранения данных, применяя к нему некоторую функцию хэширования.

Большинство хранилищ пар «ключ – значение» поддерживают только самые простые операции запроса, вставки и удаления. Чтобы частично или полностью изменить значение, приложение всегда перезаписывает существующее значение целиком. Запись больших значений занимает относительно долгое время.

Приложение может хранить в наборе значений произвольные данные, но некоторые хранилища пар «ключ – значение» накладывают ограничения на максимальный размер значений. Программное обеспечение хранилища ничего не знает о значениях, которые в нем хранятся. Все сведения о схеме поддерживаются и применяются на уровне приложения.

Хранилища пар «ключ – значение» рассчитаны на приложения, выполняющие простые операции поиска на основе значения ключа или диапазона ключей. Кроме того, хранилища пар «ключ – значение» неудобны в сценариях, где могут выполняться запросы или фильтрация по значению, а не только по ключам.

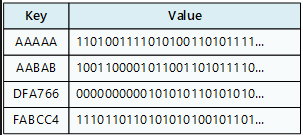


Рисунок 2 – хранилище пар «ключ – значение»

Примеры: Redis, Amazon DynamoDB, Riak.

* Хранилища данных графов

Хранилища данных графов управляют сведениями двух типов: узлами и ребрами. Узлы в этом случае представляют сущности, а ребра определяют связи между ними. Узлы и грани имеют свойства, которые предоставляют сведения о конкретном узле или грани, примерно как столбцы в реляционной таблице. Грани могут иметь направление, указывающее на характер связи.

Хранилища данных графов позволяют приложениям эффективно выполнять запросы, которые проходят через сеть узлов и ребер, а также анализировать связи между сущностями.

Например, на следующей схеме: сущности – сотрудники и отделы, грани – отношения подчинения и отдел, в котором работает каждый сотрудник.

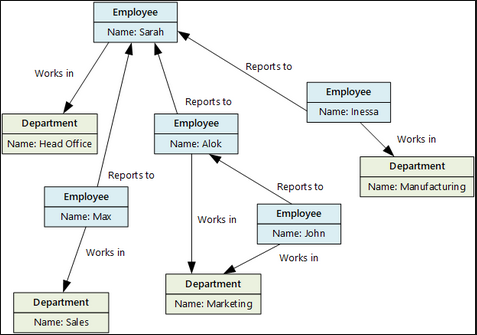


Рисунок 3 – хранилище данных графов.

Такая структура позволяет легко выполнять такие запросы, как "найти всех сотрудников, которые прямо или косвенно подчиняются Светлане" или "найти всех, кто работает в одном отделе с Дмитрием". Процессы сложного анализа выполняются быстро даже на больших графах с большим количеством сущностей и связей. Многие базы данных графов предоставляют язык запросов, который можно использовать для эффективного обхода сети связей.

Примеры: Neo4j, Amazon Neptune, JanusGraph.

* Хранилища данных объектов

Хранилища данных объектов оптимизированы для хранения и извлечения больших двоичных объектов, например изображений, текстовых файлов, видео- и аудиопотоков, объектов данных и документов приложений большого размера, образы дисков виртуальных машин. Объект состоит из сохраненных данных, метаданных и уникального идентификатора доступа к объекту. Хранилища объектов поддерживают отдельные большие файлы, а также позволяют управлять всеми файлами за счет внушительного общего объема хранилища.

Часто хранилища данных объектов используют как сетевые общие папки. Доступ к файлам, хранящимся в этих папках, можно получить через компьютерную сеть с использованием стандартных сетевых протоколов.

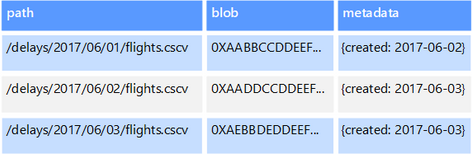


Рисунок 4 – хранилище данных объектов.

* Столбчатые хранилища данных

Столбчатое хранилище данных или хранилище семейств столбцов упорядочивает данные по столбцам и строкам. Столбчатое хранилище данных можно представить как набор табличных данных со строками и столбцами, в которых столбцы разделяются на определенные группы или семейства столбцов. Каждое семейство столбцов включает набор логически связанных столбцов, которые обычно извлекаются или управляются как единое целое. Другие данные, которые используются в других процессах, хранятся отдельно в других семействах столбцов. В семейство столбцов можно динамически добавить новые столбцы, а строки могут быть разреженными (то есть строки не обязаны иметь значение для каждого столбца).

Например, на следующей диаграмме представлен пример таблицы с двумя семействами столбцов: Identity и Contact Info:

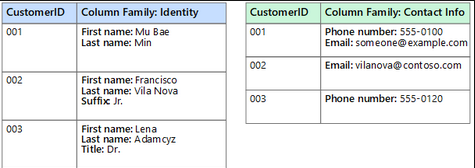


Рисунок 5 – столбчатое хранилище данных.

Данные одной сущности имеют одинаковые ключи строк во всех семействах столбцов. Такая структура, в которой строки любого объекта в семействе столбцов могут динамически изменяться, определяет важное преимущество этой категории хранилищ. Семейства столбцов очень хорошо подходят для хранения данных с различными схемами.

Примеры: Apache HBase, Amazon Redshift, Google Bigtable.

* Хранилища данных временных рядов

Данными временных рядов называются наборы значений, которые упорядочены по времени. Соответственно хранилища данных временных рядов оптимизированы для хранения данных именно такого типа. Хранилища данных временных рядов должны поддерживать очень большое число операций записи, так как обычно в них в режиме реального времени собирается большой объем данных из большого количества источников. Обновления в таких базах данных выполняются редко, а удаление чаще всего является массовой операцией.

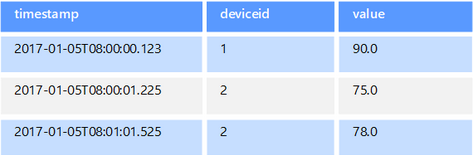


Рисунок 6 – хранилище данных временных рядов.

Размер отдельных записей в базе данных временных рядов обычно невелик, но их очень много, а значит общий размер данных быстро увеличивается. Хранилища данных временных рядов также обрабатывают данные, полученные вне очереди или несвоевременно, автоматически индексируют точки данных и оптимизируют запросы, полученные в течение определенного промежутка времени.

2 MongoDB. Проектирование схемы документа

2.1 Обоснование выбора и общая характеристика MongoDB

MongoDB - это популярная документоориентированная база данных, которая относится к семье NoSQL баз данных. MongoDB предоставляет гибкую модель хранения данных, позволяющую хранить и извлекать информацию в формате документов JSON-подобной структуры, называемых BSON (бинарное представление JSON).

MongoDB широко используется во многих сферах, включая веб-разработку, аналитику данных, управление контентом и многое другое. Его гибкая модель данных и масштабируемость делают его популярным выбором для различных приложений.

Всю модель устройства базы данных в MongoDB можно представить следующим образом:



Рисунок 7 – модель устройства MongoDB.

Если в реляционных бд содержимое составляют таблицы, то в mongodb база данных состоит из коллекций. Каждая коллекция имеет свое уникальное имя - произвольный идентификатор, состоящий из не более чем 128 различных алфавитно-цифровых символов и знака подчеркивания.

Документ представляет набор пар ключ-значение. Ключи представляют строки. Значения же могут различаться по типу данных, например, String, Array, Boolean. Binary data, Timestamp и т.д.

Для каждого документа в MongoDB определен уникальный идентификатор, который называется \_id. При добавлении документа в коллекцию данный идентификатор создается автоматически. Однако разработчик может сам явным образом задать идентификатор, а не полагаться на автоматически генерируемые, указав соответствующий ключ и его значение в документе. Данное поле должно иметь уникальное значение в рамках коллекции.

Основными причинами выбора MongoDB являются:

* Backend часть приложения будет разрабатываться с использованием платформы NodeJS, использующей язык JavaScript. MongoDB и JavaScript тесно связаны друг с другом. MongoDB предоставляет официальный драйвер для JavaScript, который позволяет взаимодействовать с базой данных MongoDB из JavaScript-приложений.
* MongoDB предоставляет официальный драйвер для Node.js, называемый MongoDB Node.js Driver. Этот драйвер позволяет взаимодействовать с MongoDB из JavaScript-приложений, разрабатываемых на платформе Node.js. Он предоставляет API для выполнения запросов, обновления данных, вставки и удаления документов, а также для управления индексами и другими аспектами базы данных.
* Mongoose – это пакет npm для Node.js, который облегчает работу с MongoDB, предоставляя объектно-документное отображение (ODM). Mongoose позволяет определить схемы данных и модели на основе MongoDB, а затем использовать их для выполнения запросов и управления данными. Mongoose также обеспечивает валидацию данных, хуки (hooks) и другие полезные функции.
* Приложение будет работать с двумя операционными системами – Windows и Linux, в которых сохраняемая информация может отличаться. Так как MongoDB позволяет хранить и обрабатывать данные различных структур и форматов внутри документов, то и хранимая информация может различаться.

2.2 Проектирование структуры документа

Для хранения информации о входе и выходе пользователей из учетных записей Windows будем использовать 2 коллекции:

1. Для хранения информации о входе пользователей;
2. Для хранения информации о выходе пользователей из учетных записей;

Для идентификации каждого компьютера будем использовать набор следующих значений: имя компьютера, MAC-адрес сетевой карты, ip-адрес компьютера. Для этого создадим следующие ключи:

* computerName – String;
* macAddress – String;
* ipAddress – String;

Также нам понадобится поле для хранения даты и времени входа или выхода:

* date – Date;

Так как в нашем приложении не нужно знать, какой именно пользователь вошел, а только дата и время входа, то для сопоставления входа пользователя с его выходом будем использовать так называемый id входа. Когда пользователь входит в учетную запись в операционной системе Windows, его сеансу назначается id, который можно получить в журнале событий. При выходе пользователя этот id также сохраняется. Из этого следует, что id входа можно использовать для сопоставления входа пользователя на определенном компьютере с его выходом из учетной записи.

Создадим соответствующее поле:

* loginId – String;

Также создадим поле для хранения аудитории, в которой находится компьютер:

* audience – String;

Следующее поле будем использовать для указания операционной системы, на которой был выполнен вход:

* operatingSystem – String;

Последнее поле будет хранить id документа:

* \_id – ObjectId;

Таким образом, конечная схема документа для хранения информации о входе и выходе пользователей из учетных записей в ОС Windows будет выглядеть следующим образом:



Рисунок 8 – схема документа.

Заключение

В данной курсовой работе был выполнен анализ основных категорий нереляционных баз данных, а также разработана схема документа для хранения информации о входе и выходе пользователей из учетных записей.

В результате проделанной работы были решены следующие задачи:

1. Проанализированы открытые источники информации, литература по теме курсовой работы.
2. Изучены основные категории нереляционных баз данных
3. Изучены особенности MongoDB.
4. Подготовлена схема документа для разрабатываемого приложения.

Созданное приложение с заданным набором функций свидетельствует о выполнении поставленных задач и о том, что цель работы достигнута.

Список использованных источников

1. Нереляционные данные и базы данных NoSQL – Режим доступа: https://learn.microsoft.com/ru-ru/azure/architecture/data-guide/big-data/non- relational-data– Дата доступа: 17.12.2023
2. Общие сведения о распределенных базах данных NoSQL – Режим доступа: <https://learn.microsoft.com/ru-ru/azure/cosmos-db/distributed-nosql?toc=https%3A%2F%2Flearn.microsoft.com%2Fru-ru%2Fazure%2Farchitecture%2Ftoc.json&bc=https%3A%2F%2Flearn.microsoft.com%2Fru-ru%2Fazure%2Farchitecture%2Fbread%2Ftoc.json> – Дата доступа: 17.12.2023
3. Введение в MongoDB – Режим доступа: <https://metanit.com/nosql/mongodb/1.1.php> – Дата доступа: 17.12.2023
4. MongoDB Documentation – Mode of access: https://www.mongodb.com/docs/ – Date of access: 17.12.2023
5. Что такое базы данных NoSQL? – Режим доступа: https://aws.amazon.com/ru/nosql/ – Дата доступа: 17.12.2023